PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-039148
(43)Date of publication of application : 12.02.1999

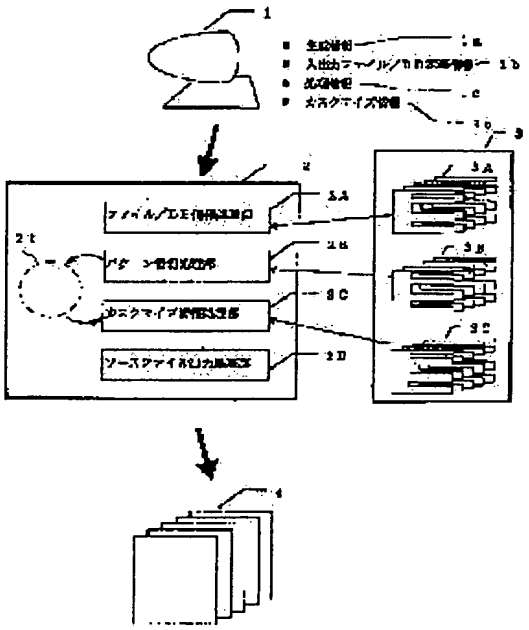
(51)Int.CI. G06F 9/06

(21)Application number : 09-198485
(22)Date of filing : 24.07.1997

(71)Applicant : HITACHI INF SYST LTD
(72)Inventor : UCHIDA MASAKO
YOSHINO SADAO

(54) AUTOMATIC SOURCE CODE GENERATION SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To integrate an existing program resource obtained by program development using a structured programming method and an object oriented programming relating to an automatic generator system of a source code effectively reusable as the existing program asset.
SOLUTION: Specification relating information for a new application is inputted to an input part 1. Based on the specification relating information, a file/DB information processing part 2A generates a source code provided with an 'object' for realizing input/output processings, a pattern information processing part 2B generates a source code provided with the 'object' corresponding to a determined processing pattern and a customization information processing part 2C generates a source code provided with the 'object' for realizing an intrinsic processing respectively. A source file output processing part 2D outputs the generated source codes respectively to prescribed source files based on the specification relative information.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

decision of rejection]
[Date of extinction of right]

* NOTICES *

Machine-generated English translation for JP11-039148

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] The chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition in various kinds of source codes obtained by development of application The chicken type database accumulated beforehand respectively possible [reuse] as existing program property, The input section which inputs various kinds of specification related information according to the specification of this application on the occasion of development of new application, The automatic generative system of the source code characterized by having the chicken type setting-out processing section which acquires the chicken type which fitted said new application from said chicken type database based on said inputted specification related information, sets up such chicken types, and generates a concrete source code.
[Claim 2] said chicken type database -- object-oriented -- the database which accumulated beforehand various kinds of chicken types based on various kinds of source codes described in the past using COBOL and COBOL -- it is -- said chicken type setting-out processing section -- said object-oriented -- the automatic generative system of the source code according to claim 1 characterized by setting up various kinds of chicken types based on various kinds of source codes described in the past using COBOL and COBOL, and generating a concrete source code.
[Claim 3] Said chicken type database is a database which accumulated beforehand the chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition which realizes predetermined radial transfer by serial access which I/O of data generates actually. Said chicken type setting-out processing section is the automatic generative system of the source code according to claim 1 or 2 characterized by generating the source code which has the object definition statement and processing instruction statement for performing predetermined radial transfer specified by said specification related information by said serial access.
[Claim 4] Said chicken type database is a database which accumulated beforehand the chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition which realizes predetermined radial transfer, without being based on said serial access. Said chicken type setting-out processing section The automatic generative system of a source code given in any 1 term of claims 1-3 characterized by generating the source code which has the object definition statement and processing instruction statement for performing without depending the predetermined radial transfer specified by said specification related information on said serial access.
[Claim 5] Said chicken type setting-out processing section determines the processing pattern which expresses the procedure by said new application based on said specification related information. The pattern-information-processing section which generates the source code which has the object definition statement and processing instruction statement corresponding to this processing pattern, Acquire the chicken type of the object definition statement of the subclass corresponding to said processing pattern determined by said pattern-information-processing section from said chicken type database, and this chicken type is concretely set up based on

said specification related information. The customize information processing section which generates the source code which has the object definition statement and processing instruction statement for realizing processing of a proper to said new application, The automatic generative system of a source code given in any 1 term of claims 1-4 characterized by having the source file output-processing section which outputs the generated source code to a predetermined source file, respectively based on said specification related information.
[Claim 6] The information storage used in order to realize the automatic generative system of the source code of a publication in any 1 term of claims 1-5 characterized by recording the database file holding said chicken type database, and the program file which realizes said chicken type setting-out processing section.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] Whichever this invention relates to the automatic generative system of a source code and it is especially developed by the technique of structured programming and object oriented programming, it is related with the information storage medium which stored the file used in order to realize the automatic generative system of the source code which can be effectively reused as existing program property, and its system.
[0002]
[Description of the Prior Art] Conventionally, high level languages, such as COBOL (Common Business Oriented Language), are known as main programming language used for development of the application program for general purpose computers. And in case it programs using such a high level language, generation of a reusable source code is effectively achieved by furthering development by the technique of structured programming as existing program property, such as a "pattern" (procedure) and "components" (mode of processing). The "components" in connection with processing centering on access to a file were mainly especially reused among the generated various kinds "components" on the occasion of development of the business oriented application of a batch-processing mold in many cases.
[0003] On the other hand, especially in recent years, the direction of furthering development instead of structured programming mentioned above by the technique of object oriented programming on the occasion of development of a client/server system is becoming general. Since the source code generated by the technique of this object oriented programming becomes reusable easily and flexibly by utilizing a "succession" function, it can aim at still more effective reuse as existing program property compared with the source code generated by the technique of structured programming. Then, the language specification is extended so that development can be furthered by the technique of object oriented programming mentioned above also about COBOL, and object-oriented COBOL which enabled reuse of the source code by the "succession" function is offered.
[0004] The "object definition automatic generation equipment" of a publication is in one of the conventional techniques which paid its attention to the "succession" function of this object-oriented COBOL at JP,8-278887,A. the record definition in the source code this conventional technique was described to be according to the language specification of object-oriented COBOL -- "succession" -- it is the technique which establishes the device made possible and enables it to reuse the existing program property still more effectively.
[0005]
[Problem(s) to be Solved by the Invention] Development is furthered by the technique of object oriented programming, and, in the case of the source code described according to the language specification of object-oriented COBOL, the greater part of this source code can be effectively reused as existing program property by applying the conventional technique of a publication to JP,8-278887,A mentioned above. However, development is furthered by the technique of structured programming and this conventional technique is not assumed at all about application to the source code described according to the language specification of conventional COBOL.

[0006] That is, let the "pattern" and the "components" which were mentioned above be the units of reuse by the technique of structured programming by the technique of object oriented programming to what encapsulated data and the behavior to this data (= "an object") being made into the unit of reuse, respectively, respectively. Among these, the utilization gestalt of "components" is a utilization gestalt of the side which itself is called and performs a certain concrete processing, and can be treated like the "object" mentioned above. For this reason, about the "components" obtained by the technique of structured programming, fusion to object oriented programming can be aimed at comparatively easily. However, the utilization gestalt of the "pattern" mentioned above is a utilization gestalt of the side which controls the flow of processing, and cannot be treated on a par with the "object" mentioned above. For this reason, about the "pattern" obtained by the technique of structured programming, there was a trouble that it was difficult to aim at fusion to object oriented programming with a gestalt as it is.
[0007] Moreover, since many of applications developed using conventional COBOL assume employment by the system which makes a general purpose computer a subject, it is not rare to take in the systems specification which requires a work file and an intermediate file. However, when such systems specification was applied to the development done by the technique of object oriented programming, if it is original, the "object" of a large number relevant to an unnecessary work file and an unnecessary intermediate file will be described as it is in a source code, and there was a trouble that decline in processing effectiveness and the ambiguity of recognition of a permanent object might be caused for unnecessary access to the work file and intermediate file by these "objects."
[0008] Therefore, the 1st object of this invention is to offer the automatic generative system of the source code which the above-mentioned trouble is solved and can be treated like [components / the "pattern" obtained by the technique of structured programming, and / both / "components"] the "object" obtained by the technique of object oriented programming.
[0009] Moreover, the 2nd object of this invention is to offer the automatic generative system of the source code which can inhibit useless access performed by the "object" relevant to the work file and intermediate file which are not recognized as a permanent object among the various kinds in the source code generated according to the systems specification of the application developed by the technique of structured programming "an object."
[0010]
[Means for Solving the Problem] In order to attain the above-mentioned object, the automatic generative system of the source code concerning claim 1 of this invention The chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition in various kinds of source codes obtained by development of application The chicken type database accumulated beforehand respectively possible [reuse] as existing program property, The input section which inputs various kinds of specification related information according to the specification of this application on the occasion of development of new application, The chicken type which fitted said new application from said chicken type database based on said inputted specification related information is acquired, and it has the chicken type setting-out processing section which sets up such chicken types and generates a concrete source code.
[0011] In invention which invention concerning claim 2 requires for above-mentioned claim 1 moreover, said chicken type database It is the database which accumulated beforehand various kinds of chicken types based on various kinds of source codes described in the past using COBOL and COBOL. object-oriented -- said chicken type setting-out processing section -- said object-oriented -- various kinds of chicken types based on various kinds of source codes described in the past using COBOL and COBOL are set up, and a concrete source code is generated.
[0012] In invention which invention concerning claim 3 requires for either above-mentioned claim 1 and claim 2 moreover, said chicken type database It is the database which accumulated beforehand the chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition which realizes predetermined radial transfer by serial access which I/O of data actually generates. Said chicken

type setting-out processing section generates the source code which has the object definition statement and processing instruction statement for performing predetermined radial transfer specified by said specification related information by said serial access.
[0013] In invention which invention concerning claim 4 requires for either above-mentioned claim 1 - claim 3 moreover, said chicken type database It is the database which accumulated beforehand the chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition which realizes predetermined radial transfer, without being based on said serial access. Said chicken type setting-out processing section generates the source code which has the object definition statement and processing instruction statement for performing without depending the predetermined radial transfer specified by said specification related information on said serial access.
[0014] In invention which invention concerning claim 5 requires for either above-mentioned claim 1 - claim 4 moreover, said chicken type setting-out processing section The processing pattern which expresses the procedure by said new application based on said specification related information is determined. The pattern-information-processing section which generates the source code which has the object definition statement and processing instruction statement corresponding to this processing pattern, Acquire the chicken type of the object definition statement of the subclass corresponding to said processing pattern determined by said pattern-information-processing section from said chicken type database, and this chicken type is concretely set up based on said specification related information. The customize information processing section which generates the source code which has the object definition statement and processing instruction statement for realizing processing of a proper to said new application, Based on said specification related information, it has the source file output-processing section which outputs the generated source code to a predetermined source file, respectively.
[0015] Moreover, the information storage used in order to realize the automatic generative system of the source code concerning claim 6 of this invention records the database file holding said chicken type database, and the program file which realizes said chicken type setting-out processing section.
[0016]
[Embodiment of the Invention] Hereafter, the gestalt of operation of the automatic generative system of the source code of this invention is explained to a detail using a drawing.
[0017] Drawing 1 is the block diagram showing 1 operation gestalt of the automatic generative system of the source code of this invention. The input section which inputs various kinds of specification related information according to the specification on the occasion of development of application with new one among this drawing, 2 acquires the chicken type suitable for new application from the chicken type database which following-** based on the inputted specification related information. The chicken type setting-out processing section which sets up such chicken types and generates a concrete source code, 3 the chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition in various kinds of source codes obtained by development of application The chicken type database accumulated beforehand respectively possible [reuse] as existing program property is shown, respectively.
[0018] moreover 2A by serial access which I/O of data generates actually Predetermined radial transfer The predetermined object definition and this predetermined object definition to realize It is based on the chicken type which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition which realizes predetermined radial transfer, without being based on the chicken type or serial access accumulated into the chicken type database 3 which consists of a predetermined program definition to treat. Predetermined radial transfer The file / DB information processing section which generates the concrete source code to perform, 2B determines the processing pattern which expresses the procedure by new application based on specification related information. The pattern-information-processing section which generates the source code which has the object definition statement and processing instruction statement corresponding to this

processing pattern, 2C acquires the chicken type of the object definition statement of the subclass corresponding to processing pattern 2t determined by pattern-information-processing section 2B from the chicken type database 3, and sets up this chicken type concretely based on specification related information. The customize information processing section which generates the source code which has the object definition statement and processing instruction statement for realizing processing of a proper to new application, The source file output-processing section which outputs the source code by which 2D was generated based on specification related information to a predetermined source file, respectively, ****** of the source code relevant to the processing pattern called for by pattern-information-processing section 2B 2t, and the input/output operation from which 3A becomes an object for acquisition by the file / DB information processing section 2A, ****** of the source code relevant to the procedure of the whole application with which 3B becomes an object for acquisition by pattern-information-processing section 2B, ****** of the source code relevant to mode of processing performed by application every place from which 3C serves as an object for acquisition by customize information processing section 2C, and 4 show the source file group containing the source code generated by the automatic generative system of a source code mentioned above, respectively.

[0019] In drawing 1 , each information on creation information 1a, a I / O file / DB definition information 1b, processing information 1c, and 1d of customize information is inputted into the input section 1 on the occasion of new application development as various kinds of specification related information according to the specification of this application.

[0020] Then, the file / DB information processing section 2A of the chicken type setting-out processing section 2 The I / O file / DB definition information 1b in specification related information, Or it is based on the chicken type accumulated into the chicken type database 3 which consists of a predetermined program definition treating the predetermined object definition and this predetermined object definition which realizes predetermined radial transfer, without being based on serial access. the serial access mentioned above -- The source code which has the object definition statement (the file / DB actuation object definition statement) and processing instruction statement for performing predetermined radial transfer specified by the I / O file / DB definition information 1b is generated. Pattern-information-processing section 2B of the chicken type setting-out processing section 2 determines processing pattern 2t which expresses the procedure by new application based on processing information 1c, and generates the source code which has the object definition statement and processing instruction statement corresponding to this processing pattern 2t. Customize information processing section 2C of the chicken type setting-out processing section 2 acquires the chicken type of the object definition statement of the subclass corresponding to processing pattern 2t determined by pattern-information-processing section 2B from the chicken type database 3, sets up this chicken type concretely based on 1d of customize information, and generates the source code which has the object definition statement and processing instruction statement for realizing processing of a proper to new application. Source file output-processing section 2D of the chicken type setting-out processing section 2 outputs the generated source code to the predetermined source file 4 based on creation information 1a, respectively.

[0021] Drawing 2 is a flow chart explaining the flow of processing by the file / DB information processing section 2A in drawing 1 , and drawing 3 is drawing showing a chicken type example of the source code set up by processing of drawing 2 . Here, the case where a file is set as the object of radial transfer is explained.

[0022] In drawing 2 , the I / O file / DB definition information 1b for one file are acquired from the specification related information first inputted in the input section 1 (step 201). This I / O file / DB definition information 1b include assignment information, such as a file name, a file stereo name, file organization, a record name, a record definition, and key information. Then, with reference to the chicken type database 3, the chicken type of the file / record definition sentence corresponding to the file organization in assignment information is acquired (step 202), and assignment information is set as the acquired chicken type (step 203). Moreover, the chicken type of the file / record actuation sentence corresponding to the same file organization is acquired (step 204), and assignment information is set as the acquired chicken type (step 205).

For example, as shown in drawing 3 , at step 203, a file name 311, file organization 312, and a key item 313 (assignment information) are set as the define the file mold 310 (chicken type of a file / record definition sentence), a file name 321, a record name 322, and a record definition 323 (assignment information) are set as the record definition mold 320 (chicken type of a file / record definition sentence) for the source code of a define the file, and the source code of a record definition is generated, respectively. Moreover, at step 205, a file name 331 (assignment information) is set as a file / record actuation sentence type 330 (chicken type of a file / record actuation sentence), and the source code of a file / record actuation sentence is generated.
[0023] Next, with reference to the chicken type database 3, chicken type 300 of the source code of the object definition corresponding to the file organization in the assignment information mentioned above is acquired (step 206). The source code of the file / record actuation sentence generated at the source code and step 205 of a file / record definition sentence which were generated at step 203 is set as the acquired chicken type. By this The source code which has the object definition statement and processing instruction statement for desired file / record actuation is generated (step 207). It judges whether there is any other file used at the end (step 208), and only the number of the files to be used repeats and performs processing of steps 201-207.
[0024] Drawing 4 is a flow chart explaining the flow of processing by pattern-information-processing section 2B in drawing 1 , and drawing 5 is drawing showing a chicken type example of the source code set up by processing of drawing 4 .
[0025] In drawing 4 , the input situation of processing information 1c inputted in the input section 1 is checked first (step 401), for example, it asks for the number of I / O files, master transaction classification, processing key assignment of an input file, etc., and judges whether the decision of a desired processing pattern is possible only at the present processing information 1c (step 402). Consequently, only by the present processing information 1c, when a processing pattern cannot be determined (step 402= No), the input request of the lack information which is needed for the decision of a desired processing pattern is performed (step 403), and it returns to step 401 again. On the other hand, when the decision of a desired processing pattern is possible only at the present processing information 1c (step 402= Yes), in order to use it by customize information processing section 2C, determined processing pattern 2t is stored (step 404). Next, with reference to the chicken type database 3, the chicken type of the source file of the object definition corresponding to determined processing pattern 2t is acquired (step 405). And the information specified as the chicken type of the acquired source file by processing information 1c is set up (step 406). For example, as shown in drawing 5 , when chicken type 500 of the object definition to which the processing pattern determined at step 404 is "matching", and corresponds at step 405 is acquired, at step 406, the conditional statement 511 for the processing key judging which is the information specified as the processing description part 510 in this chicken type 500 by processing information 1c is set up. After this, further, with reference to the chicken type database 3, the chicken type of a program definition of a main program is acquired (step 407), the control processing description for object definition control, for example, description of setting-out processing of the handle of an activity file etc., is added to this chicken type, and the source code which has the object definition statement and processing instruction statement corresponding to the above-mentioned processing pattern by this is generated (step 408).
[0026] Drawing 6 is a flow chart explaining the flow of processing by customize information processing section 2C in drawing 1 , and drawing 7 is drawing showing a chicken type example of the source code set up by processing of drawing 6 .
[0027] In drawing 6 , processing pattern 2t first stored by processing of step 404 by pattern-information-processing section 2B mentioned above is acquired (step 601). Next, with reference to the chicken type database 3, chicken type 700 of an object definition of the subclass corresponding to this processing pattern 2t is acquired (step 602), and it asks for the processing for customize of the proper specified as 1d of customize information about this chicken type 700. Both Anh both [either or] which perform processing for customize here when match processing and the key value which will be performed when a key value matches if it is the thing

of the concrete processing performed when the flow of processing agrees to a specific processing pattern, for example, a processing pattern is "matching" do not match can turn into processing for customize. Finally, the description 711 of the processing for customize for which the method description corresponding to applicable processing of chicken type 700 of an object definition of the subclass acquired at step 602 was asked is set up, and the source code which has object definition statement and processing instruction statement for this to realize processing (match processing) of a proper is generated (step 604).

[0028] Drawing 8 is a flow chart explaining the flow of processing by source file output-processing section 2D in drawing 1 . Creation information 1a is acquired from the specification related information inputted into the beginning in the input section 1 among this drawing (step 801). Here, creation information 1a is for example, a generation place directory file name. Then, the class name and program name in creation information 1a are set as CLASS-ID and PROGRAM-ID in a file / DB information processing section 2A, pattern-information-processing section 2B, and the source code previously generated by customize information processing section 2C, respectively (step 802). Moreover, the file name of the source file which stores each source code is assembled using these CLASS-ID and PROGRAM-ID (step 803). The source code corresponding to the source file which added the assembled file name at the end, respectively is outputted (step 804).

[0029] Drawing 9 is drawing showing an example of two or more source files containing the source code generated by the system of drawing 1 , and expresses the source file group which constitutes the application which performs direct access to a work file or an intermediate file. The generated source file among this drawing Predetermined radial transfer by serial access The object definition statement and processing instruction statement for performing the source code, i.e., the file information management class, which it has To the source file 92 containing the source code, i.e., the pattern processing management class, which has the object definition statement and processing instruction statement corresponding to the included source file 91 and the processing pattern showing the procedure by new application, and new application, processing of a proper The object definition statement and processing instruction statement for realizing the source code, i.e., the customize processing management class, which it has It is divided roughly into a total of four kinds of source files of the source file 94 containing the source code, i.e., the main program, which has the processing instruction statement of the program definition which summarizes the included source files 93 and these source files 91-93.

[0030] Drawing 10 is drawing showing other examples of two or more source files containing the source code generated by the system of drawing 1 , and expresses the source file group which constitutes the application which performs direct access neither to a work file nor an intermediate file. The generated source file among this drawing Predetermined radial transfer The procedure by the source file 102 containing the source file 101 and element class containing the source code, i.e., the collection class, which has the object definition statement and processing instruction statement for performing without being based on serial access, and new application The object definition statement and processing instruction statement for realizing processing of a proper to the source file 103 containing the source code, i.e., the pattern processing management class, which has the object definition statement and processing instruction statement corresponding to the processing pattern which expresses, and new application the source code, i.e., the customize processing management class, which it has It is divided roughly into a total of five kinds of source files of the source file 105 containing the source code, i.e., the main program, which has the processing instruction statement of the program definition which summarizes the included source files 104 and these source files 101-104.

[0031] As mentioned above, according to the automatic generative system of the source code of this operation gestalt The specification related information which specifies the "pattern" and the "components" which were obtained on the occasion of new application development by the technique of the conventional structured programming which it is going to reuse with this new application, Namely, by deciding appropriately each information on creation information 1a, a I / O file / DB definition information 1b, processing information 1c, and 1d of customize information, and inputting into the input section 1 The chicken type corresponding to the corresponding

"pattern" and the "components" is acquired from the chicken type database 3. About "a pattern, i.e., a processing pattern," While the input request of lack information required for the decision is performed by pattern-information-processing section 2B, description of the processing for customize of a proper is set as new application by customize information processing section 2C at a chicken type. For this reason, it becomes unnecessary to distinguish intentionally the "object" obtained by the technique of the "pattern" and the "components" which were obtained by the technique of the conventional structured programming, and object oriented programming, and can treat similarly, and fusion to the existing program property and the object oriented programming which were obtained by the program development by the technique of structured programming until now can be aimed at easily.

[0032] Moreover, it becomes generable [the source file 102 containing the source file 101 and the element class containing the source code, i.e., the collection class, which has the object definition statement and the processing instruction statement for performing without depending predetermined radial transfer on serial access as shown in drawing 10] by accumulating beforehand the chicken type which consists of a predetermined program definition treating the object definition which realizes radial transfer, without accessing serially to a work file or an intermediate file, and this object definition into the chicken type database 3. For this reason, useless access performed by the "object" relevant to the work file and intermediate file which are not recognized as a permanent object is inhibited, and decline in processing effectiveness and generating of the ambiguity of recognition of a permanent object can be prevented.

[0033] In addition, the database file holding the chicken type database 3 mentioned above and the program file which realizes the chicken type setting-out processing section 2 can be offered in the form kept by information storages, such as a floppy disk and CD-ROM. In this case, even if it keeps both collectively to the same information storage medium, both may be separately kept to a different information storage medium, or whichever is sufficient.

[0034]

[Effect of the Invention] As explained in detail above, according to the automatic generative system of the source code of this invention, it becomes unnecessary to distinguish intentionally the "object" obtained by the technique of the "pattern" and the "components" which were obtained by the technique of the conventional structured programming, and object oriented programming, and can treat similarly, and fusion to the existing program property and the object oriented programming which were obtained by the program development by the technique of structured programming until now can be aimed at easily.

[0035] Moreover, if the chicken type of the "object" which realizes radial transfer without serial access is beforehand accumulated into the chicken type database, useless access performed by the "object" relevant to the work file and intermediate file which are not recognized as a permanent object is inhibited, and decline in processing effectiveness and generating of the ambiguity of recognition of a permanent object can be prevented.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] It is the block diagram showing 1 operation gestalt of the automatic generative system of the source code of this invention.
[Drawing 2] It is a flow chart explaining the flow of processing by the file / DB information processing section in drawing 1 .
[Drawing 3] It is drawing showing a chicken type example of the source code set up by processing of drawing 2 .
[Drawing 4] It is a flow chart explaining the flow of processing by the pattern-information-processing section in drawing 1 .
[Drawing 5] It is drawing showing a chicken type example of the source code set up by processing of drawing 4 .
[Drawing 6] It is a flow chart explaining the flow of processing by the customize information processing section in drawing 1 .
[Drawing 7] It is drawing showing a chicken type example of the source code set up by processing of drawing 6 .
[Drawing 8] It is a flow chart explaining the flow of processing by the source file output-processing section in drawing 1 .
[Drawing 9] It is drawing showing an example of two or more source files containing the source code generated by the system of drawing 1 .
[Drawing 10] It is drawing showing other examples of two or more source files containing the source code generated by the system of drawing 1 .
[Description of Notations]
1 Input Section
2 Chicken Type Setting-Out Processing Section
3 Chicken Type Database
2A A file / DB information processing section
2B Pattern-information-processing section
2C Customize information processing section
2D Source file output-processing section
2t Recognized processing pattern
3A ****** of the source code relevant to input/output operation
3B ****** of the source code relevant to the procedure of the whole application
3C ****** of the source code relevant to mode of processing performed by application every place
4 Source File Group Containing Generated Source Code

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－39148

(43)公開日　平成11年(1999)２月12日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/06	５３０	Ｇ０６Ｆ　9/06	５３０Ｖ
			５３０Ｗ

審査請求　未請求　請求項の数６　ＯＬ　（全 13 頁）

(21)出願番号　特願平9－198485

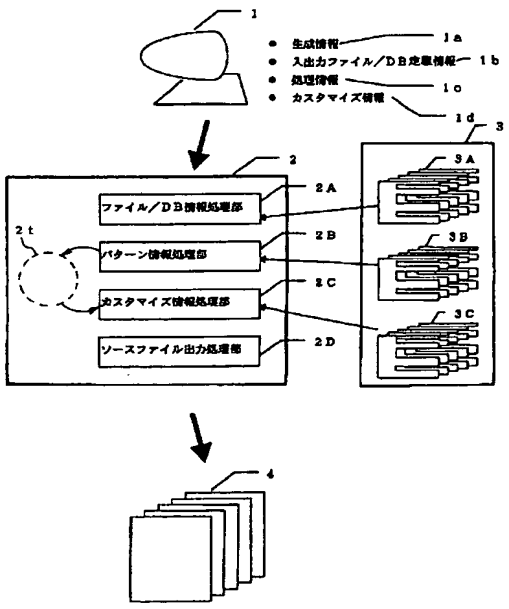
(22)出願日　平成９年(1997)７月24日

(71)出願人　000152985
株式会社日立情報システムズ
東京都渋谷区道玄坂１丁目16番５号
(72)発明者　内田　雅子
東京都渋谷区道玄坂一丁目16番５号　株式会社日立情報システムズ内
(72)発明者　吉野　貞夫
東京都渋谷区道玄坂一丁目16番５号　株式会社日立情報システムズ内
(74)代理人　弁理士　武　顕次郎

(54)【発明の名称】　ソースコードの自動生成システム

(57)【要約】
【課題】　既存のプログラム資産として有効に再利用可能なソースコードの自動生成システムに関し、構造化プログラミングの手法によるプログラム開発で得られた既存のプログラム資産とオブジェクト指向プログラミングとの融合を図る。
【解決手段】　新たなアプリケーションについての仕様関連情報を入力部１に入力する。この仕様関連情報に基づいて、ファイル／ＤＢ情報処理部２Ａは入出力処理を実現する“オブジェクト”を有するソースコードを、パターン情報処理部２Ｂは決定される処理パターンに対応する“オブジェクト”を有するソースコードを、カスタマイズ情報処理部２Ｃは固有の処理を実現するための“オブジェクト”を有するソースコードを、それぞれ生成する。ソースファイル出力処理部２Ｄは、仕様関連情報に基づき、生成されたソースコードをそれぞれ所定のソースファイルに出力する。

【図１】



1

【特許請求の範囲】

【請求項１】　アプリケーションの開発で得られた各種のソースコード内の所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型を、それぞれ既存のプログラム資産として再利用可能にあらかじめ蓄積した雛型データベースと、

新たなアプリケーションの開発に際し、このアプリケーションの仕様に応じた各種の仕様関連情報を入力する入力部と、

入力された前記仕様関連情報に基づいて前記雛型データベースから前記新たなアプリケーションに適した雛型を取得し、これらの雛型を設定して具体的なソースコードを生成する雛型設定処理部とを備えたことを特徴とするソースコードの自動生成システム。

【請求項２】　前記雛型データベースは、オブジェクト指向ＣＯＢＯＬ及びＣＯＢＯＬを用いて過去に記述された各種のソースコードに基づく各種の雛型をあらかじめ蓄積したデータベースであって、

前記雛型設定処理部は、前記オブジェクト指向ＣＯＢＯＬ及びＣＯＢＯＬを用いて過去に記述された各種のソースコードに基づく各種の雛型を設定して具体的なソースコードを生成することを特徴とする請求項１記載のソースコードの自動生成システム。

【請求項３】　前記雛型データベースは、実際にデータの入出力が発生する逐次アクセスによって所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型をあらかじめ蓄積したデータベースであって、

前記雛型設定処理部は、前記仕様関連情報で指定された所定の入出力処理を前記逐次アクセスによって実行するためのオブジェクト定義文及び処理命令文を有するソースコードを生成することを特徴とする請求項１または２記載のソースコードの自動生成システム。

【請求項４】　前記雛型データベースは、前記逐次アクセスによることなく所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型をあらかじめ蓄積したデータベースであって、

前記雛型設定処理部は、前記仕様関連情報で指定された所定の入出力処理を前記逐次アクセスによることなく実行するためのオブジェクト定義文及び処理命令文を有するソースコードを生成することを特徴とする請求項１～３のいずれか一項に記載のソースコードの自動生成システム。

【請求項５】　前記雛型設定処理部は、

前記仕様関連情報に基づいて前記新たなアプリケーションによる処理手順を表す処理パターンを決定し、この処理パターンに対応するオブジェクト定義文及び処理命令文を有するソースコードを生成するパターン情報処理部と、

2

前記パターン情報処理部によって決定された前記処理パターンに対応するサブクラスのオブジェクト定義文の雛型を前記雛型データベースから取得し、この雛型を前記仕様関連情報に基づいて具体的に設定して、前記新たなアプリケーションに固有の処理を実現するためのオブジェクト定義文及び処理命令文を有するソースコードを生成するカスタマイズ情報処理部と、

前記仕様関連情報に基づき、生成されたソースコードをそれぞれ所定のソースファイルに出力するソースファイル出力処理部とを有することを特徴とする請求項１～４のいずれか一項に記載のソースコードの自動生成システム。

【請求項６】　前記雛型データベースを保持するデータベースファイルと、前記雛型設定処理部を実現させるプログラムファイルとを記録したことを特徴とする請求項１～５のいずれか一項に記載のソースコードの自動生成システムを実現するために使用する情報記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はソースコードの自動生成システムに係り、特に、構造化プログラミング，オブジェクト指向プログラミングのどちらの手法で開発されたものであっても、既存のプログラム資産として有効に再利用することが可能なソースコードの自動生成システム及びそのシステムを実現するために使用するファイルを格納した情報記憶媒体に関する。

【０００２】

【従来の技術】従来、汎用コンピュータ用のアプリケーションプログラムの開発に利用される主要なプログラミング言語として、例えばＣＯＢＯＬ（ Common Business Oriented Language）などの高級言語が知られている。そして、このような高級言語を用いてプログラミングを行う際は、構造化プログラミングという手法で開発作業を進めることにより、“パターン”（処理手順）や“部品”（処理方式）などの既存のプログラム資産として有効に再利用可能なソースコードの生成が図られている。生成された各種“部品”のうち、主にファイルへのアクセスを中心とした処理に関わる“部品”は、特にバッチ処理型の業務用アプリケーションの開発に際して再利用されることが多かった。

【０００３】一方、近年では、特にクライアントサーバシステムの開発に際して、上述した構造化プログラミングの代わりにオブジェクト指向プログラミングという手法で開発作業を進めることの方が一般的になりつつある。このオブジェクト指向プログラミングの手法で生成されるソースコードは、“継承”機能を活用することにより容易かつ柔軟に再利用可能となるため、構造化プログラミングの手法で生成されるソースコードにくらべると、既存のプログラム資産としてさらに有効な再利用を図ることができる。そこで、ＣＯＢＯＬについても上述

したオブジェクト指向プログラミングの手法で開発作業を進めることができるようにその言語仕様を拡張し、“継承”機能によるソースコードの再利用を可能としたオブジェクト指向ＣＯＢＯＬが提供されている。

【０００４】このオブジェクト指向ＣＯＢＯＬの“継承”機能に着目した従来技術のひとつに、特開平８－２７８８８７号公報に記載の「オブジェクト定義自動生成装置」がある。この従来技術は、オブジェクト指向ＣＯＢＯＬの言語仕様にしたがって記述されたソースコード中のレコード定義を“継承”可能とする機構を設け、既存のプログラム資産をさらに有効に再利用できるようにする技術である。

【０００５】

【発明が解決しようとする課題】オブジェクト指向プログラミングの手法で開発作業が進められ、オブジェクト指向ＣＯＢＯＬの言語仕様にしたがって記述されたソースコードの場合、上述した特開平８－２７８８８７号公報に記載の従来技術を適用することにより、このソースコードの大部分を既存のプログラム資産として有効に再利用することができる。しかしながら、この従来技術は、構造化プログラミングの手法で開発作業が進められ、旧来のＣＯＢＯＬの言語仕様にしたがって記述されたソースコードへの適用について何ら想定していない。

【０００６】すなわち、オブジェクト指向プログラミングの手法では、データ及びこのデータに対する振る舞いをカプセル化したもの（＝“オブジェクト”）がそれぞれ再利用の単位とされるのに対し、構造化プログラミングの手法では、上述した“パターン”及び“部品”がそれぞれ再利用の単位とされる。このうち、“部品”の利用形態は、それ自身が呼び出されて何らかの具体的な処理を実行する側の利用形態であって、上述した“オブジェクト”と同様に扱うことが可能である。このため、構造化プログラミングの手法で得られた“部品”については比較的容易にオブジェクト指向プログラミングとの融合を図ることができる。しかしながら、上述した“パターン”の利用形態は、処理の流れを制御する側の利用形態であって、上述した“オブジェクト”と同等に扱うことはできない。このため、構造化プログラミングの手法で得られた“パターン”についてはそのままの形態でオブジェクト指向プログラミングとの融合を図ることが困難であるという問題点があった。

【０００７】また、旧来のＣＯＢＯＬを用いて開発されたアプリケーションの多くは、汎用コンピュータを主体とするシステムでの運用を想定しているため、作業ファイルや中間ファイルを要するシステム仕様が取り入れられていることが少なくない。ところが、このようなシステム仕様をオブジェクト指向プログラミングの手法で進められる開発作業に適用すると、本来なら不要な作業ファイルや中間ファイルに関連する多数の“オブジェクト”をソースコード中にそのまま記述することとなり、

これらの“オブジェクト”による作業ファイルや中間ファイルへの不要なアクセスのために処理効率の低下や永続オブジェクトの認識のあいまいさを招いてしまうことがあるという問題点があった。

【０００８】したがって本発明の第１の目的は、上記の問題点を解決して、構造化プログラミングの手法で得られた“パターン”及び“部品”のどちらについても、オブジェクト指向プログラミングの手法で得られた“オブジェクト”と同様に扱うことが可能なソースコードの自動生成システムを提供することにある。

【０００９】また、本発明の第２の目的は、構造化プログラミングの手法で開発されたアプリケーションのシステム仕様にしたがって生成されたソースコード中の各種“オブジェクト”のうち、永続オブジェクトとして認識されない作業ファイルや中間ファイルに関連する“オブジェクト”によって行われる無駄なアクセスの抑止が可能なソースコードの自動生成システムを提供することにある。

【００１０】

【課題を解決するための手段】上記の目的を達成するため、本発明の請求項１に係るソースコードの自動生成システムは、アプリケーションの開発で得られた各種のソースコード内の所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型を、それぞれ既存のプログラム資産として再利用可能にあらかじめ蓄積した雛型データベースと、新たなアプリケーションの開発に際し、このアプリケーションの仕様に応じた各種の仕様関連情報を入力する入力部と、入力された前記仕様関連情報に基づいて前記雛型データベースから前記新たなアプリケーションに適した雛型を取得し、これらの雛型を設定して具体的なソースコードを生成する雛型設定処理部とを備えたものである。

【００１１】また、請求項２に係る発明は、上記請求項１に係る発明において、前記雛型データベースは、オブジェクト指向ＣＯＢＯＬ及びＣＯＢＯＬを用いて過去に記述された各種のソースコードに基づく各種の雛型をあらかじめ蓄積したデータベースであって、前記雛型設定処理部は、前記オブジェクト指向ＣＯＢＯＬ及びＣＯＢＯＬを用いて過去に記述された各種のソースコードに基づく各種の雛型を設定して具体的なソースコードを生成するものである。

【００１２】また、請求項３に係る発明は、上記請求項１及び請求項２のいずれかに係る発明において、前記雛型データベースは、実際にデータの入出力が発生する逐次アクセスによって所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型をあらかじめ蓄積したデータベースであって、前記雛型設定処理部は、前記仕様関連情報で指定された所定の入出力処理を前記逐次アクセスによって実行するためのオブジェクト定義文及び処

理命令文を有するソースコードを生成するものである。

【００１３】また、請求項４に係る発明は、上記請求項１～請求項３のいずれかに係る発明において、前記雛型データベースは、前記逐次アクセスによることなく所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型をあらかじめ蓄積したデータベースであって、前記雛型設定処理部は、前記仕様関連情報で指定された所定の入出力処理を前記逐次アクセスによることなく実行するためのオブジェクト定義文及び処理命令文を有するソースコードを生成するものである。

【００１４】また、請求項５に係る発明は、上記請求項１～請求項４のいずれかに係る発明において、前記雛型設定処理部は、前記仕様関連情報に基づいて前記新たなアプリケーションによる処理手順を表す処理パターンを決定し、この処理パターンに対応するオブジェクト定義文及び処理命令文を有するソースコードを生成するパターン情報処理部と、前記パターン情報処理部によって決定された前記処理パターンに対応するサブクラスのオブジェクト定義文の雛型を前記雛型データベースから取得し、この雛型を前記仕様関連情報に基づいて具体的に設定して、前記新たなアプリケーションに固有の処理を実現するためのオブジェクト定義文及び処理命令文を有するソースコードを生成するカスタマイズ情報処理部と、前記仕様関連情報に基づき、生成されたソースコードをそれぞれ所定のソースファイルに出力するソースファイル出力処理部とを有するものである。

【００１５】また、本発明の請求項６に係るソースコードの自動生成システムを実現するために使用する情報記憶媒体は、前記雛型データベースを保持するデータベースファイルと、前記雛型設定処理部を実現させるプログラムファイルとを記録したものである。

【００１６】

【発明の実施の形態】以下、本発明のソースコードの自動生成システムの実施の形態を図面を用いて詳細に説明する。

【００１７】図１は、本発明のソースコードの自動生成システムの一実施形態を示すブロック図である。同図中、１は新たなアプリケーションの開発に際してその仕様に応じた各種の仕様関連情報を入力する入力部、２は入力された仕様関連情報に基づいて次述する雛型データベースから新たなアプリケーションに適した雛型を取得し、これらの雛型を設定して具体的なソースコードを生成する雛型設定処理部、３はアプリケーションの開発で得られた各種のソースコード内の所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型を、それぞれ既存のプログラム資産として再利用可能にあらかじめ蓄積した雛型データベースを、それぞれ示す。

【００１８】また、２Ａは実際にデータの入出力が発生

する逐次アクセスによって所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型データベース３中に蓄積された雛型あるいは逐次アクセスによることなく所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型に基づいて所定の入出力処理を実行する具体的なソースコードを生成するファイル／ＤＢ情報処理部、２Ｂは仕様関連情報に基づいて新たなアプリケーションによる処理手順を表す処理パターンを決定し、この処理パターンに対応するオブジェクト定義文及び処理命令文を有するソースコードを生成するパターン情報処理部、２Ｃはパターン情報処理部２Ｂによって決定された処理パターン２ｔに対応するサブクラスのオブジェクト定義文の雛型を雛型データベース３から取得し、この雛型を仕様関連情報に基づいて具体的に設定して、新たなアプリケーションに固有の処理を実現するためのオブジェクト定義文及び処理命令文を有するソースコードを生成するカスタマイズ情報処理部、２Ｄは仕様関連情報に基づき、生成されたソースコードをそれぞれ所定のソースファイルに出力するソースファイル出力処理部、２ｔはパターン情報処理部２Ｂにより求められた処理パターン、３Ａはファイル／ＤＢ情報処理部２Ａによる取得対象となる入出力操作に関連したソースコードの雛型群、３Ｂはパターン情報処理部２Ｂによる取得対象となるアプリケーション全体の処理手順に関連したソースコードの雛型群、３Ｃはカスタマイズ情報処理部２Ｃによる取得対象となるアプリケーション各所で実行される処理方式に関連したソースコードの雛型群、４は上述したソースコードの自動生成システムによって生成されたソースコードを含むソースファイル群を、それぞれ示す。

【００１９】図１において、新たなアプリケーション開発に際し、このアプリケーションの仕様に応じた各種の仕様関連情報として、生成情報１ａ，入出力ファイル／ＤＢ定義情報１ｂ，処理情報１ｃ，カスタマイズ情報１ｄの各情報を、入力部１に入力する。

【００２０】続いて、雛型設定処理部２のファイル／ＤＢ情報処理部２Ａは、仕様関連情報中の入出力ファイル／ＤＢ定義情報１ｂと、上述した逐次アクセスによってあるいは逐次アクセスによらずに所定の入出力処理を実現する所定のオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型データベース３中に蓄積された雛型とに基づき、入出力ファイル／ＤＢ定義情報１ｂで指定された所定の入出力処理を実行するためのオブジェクト定義文（ファイル／ＤＢ操作オブジェクト定義文）及び処理命令文を有するソースコードを生成する。雛型設定処理部２のパターン情報処理部２Ｂは、処理情報１ｃに基づいて新たなアプリケーションによる処理手順を表す処理パターン２ｔを決定し、この処理パターン２ｔに対応するオブジェクト定義文及び

処理命令文を有するソースコードを生成する。雛型設定処理部２のカスタマイズ情報処理部２Ｃは、パターン情報処理部２Ｂによって決定された処理パターン２ｔに対応するサブクラスのオブジェクト定義文の雛型を雛型データベース３から取得し、この雛型をカスタマイズ情報１ｄに基づいて具体的に設定して、新たなアプリケーションに固有の処理を実現するためのオブジェクト定義文及び処理命令文を有するソースコードを生成する。雛型設定処理部２のソースファイル出力処理部２Ｄは、生成情報１ａに基づき、生成されたソースコードをそれぞれ所定のソースファイル４に出力する。

【００２１】図２は、図１中のファイル／ＤＢ情報処理部２Ａによる処理の流れを説明するフローチャートであり、図３は、図２の処理によって設定されるソースコードの雛型の一例を示す図である。ここでは、ファイルを入出力処理の対象とする場合について説明する。

【００２２】図２において、最初に入力部１で入力された仕様関連情報から、１ファイル分の入出力ファイル／ＤＢ定義情報１ｂを取得する（ステップ２０１）。この入出力ファイル／ＤＢ定義情報１ｂは、例えば、ファイル名，ファイル実体名，ファイル編成，レコード名称，レコード定義，キー情報などの指定情報を含む。続いて、雛型データベース３を参照し、指定情報中のファイル編成に対応するファイル／レコード定義文の雛型を取得して（ステップ２０２）、取得した雛型に指定情報を設定する（ステップ２０３）。また、同じファイル編成に対応するファイル／レコード操作文の雛型を取得して（ステップ２０４）、取得した雛型に指定情報を設定する（ステップ２０５）。例えば図３に示すように、ステップ２０３では、ファイル名３１１，ファイル編成３１２，キー項目３１３（指定情報）をファイル定義型３１０（ファイル／レコード定義文の雛型）に設定してファイル定義のソースコードを、ファイル名３２１，レコード名３２２，レコード定義３２３（指定情報）をレコード定義型３２０（ファイル／レコード定義文の雛型）に設定してレコード定義のソースコードを、それぞれ生成する。また、ステップ２０５では、ファイル名３３１（指定情報）をファイル／レコード操作文型３３０（ファイル／レコード操作文の雛型）に設定して、ファイル／レコード操作文のソースコードを生成する。

【００２３】次に、雛型データベース３を参照して、上述した指定情報中のファイル編成に対応したオブジェクト定義のソースコードの雛型３００を取得して（ステップ２０６）、取得した雛型に、ステップ２０３で生成したファイル／レコード定義文のソースコード及びステップ２０５で生成したファイル／レコード操作文のソースコードを設定し、これによって、所望のファイル／レコード操作のためのオブジェクト定義文及び処理命令文を有するソースコードを生成する（ステップ２０７）。最後に使用するファイルが他にあるか否か判定し（ステップ２０８）、使用するファイルの数だけステップ２０１～２０７の処理を繰り返し実行する。

【００２４】図４は、図１中のパターン情報処理部２Ｂによる処理の流れを説明するフローチャートであり、図５は、図４の処理によって設定されるソースコードの雛型の一例を示す図である。

【００２５】図４において、最初に、入力部１で入力された処理情報１ｃの入力状況をチェックし（ステップ４０１）、例えば入出力ファイル数や入力ファイルのマスタ・トランザクション種別及び処理キー指定などを求めて、現状の処理情報１ｃのみで所望の処理パターンの決定が可能か否かを判定する（ステップ４０２）。この結果、現状の処理情報１ｃのみでは処理パターンを決定できない場合（ステップ４０２＝Ｎｏ）、所望の処理パターンの決定に必要となる不足情報の入力要求を行って（ステップ４０３）、再びステップ４０１に戻る。一方、現状の処理情報１ｃのみで所望の処理パターンの決定が可能であった場合（ステップ４０２＝Ｙｅｓ）、カスタマイズ情報処理部２Ｃで使用するため、決定された処理パターン２ｔを格納しておく（ステップ４０４）。次に雛型データベース３を参照して、決定された処理パターン２ｔに対応したオブジェクト定義のソースファイルの雛型を取得する（ステップ４０５）。そして、取得したソースファイルの雛型に、処理情報１ｃで指定された情報を設定する（ステップ４０６）。例えば図５に示すように、ステップ４０４で決定された処理パターンが“マッチング”であり、ステップ４０５で対応するオブジェクト定義の雛型５００を取得した場合、ステップ４０６では、この雛型５００中の処理記述部分５１０に、処理情報１ｃで指定された情報である処理キー判定のための条件文５１１を設定する。この後さらに、雛型データベース３を参照して、メインプログラムのプログラム定義の雛型を取得し（ステップ４０７）、この雛型にオブジェクト定義制御のための制御処理記述、例えば使用ファイルのハンドルの設定処理などの記述を追加し、これによって上記処理パターンに対応するオブジェクト定義文及び処理命令文を有するソースコードを生成する（ステップ４０８）。

【００２６】図６は、図１中のカスタマイズ情報処理部２Ｃによる処理の流れを説明するフローチャートであり、図７は、図６の処理によって設定されるソースコードの雛型の一例を示す図である。

【００２７】図６において、最初に、上述したパターン情報処理部２Ｂによるステップ４０４の処理で格納された処理パターン２ｔを取得する（ステップ６０１）。次に、雛型データベース３を参照して、この処理パターン２ｔに対応したサブクラスのオブジェクト定義の雛型７００を取得し（ステップ６０２）、この雛型７００についてカスタマイズ情報１ｄに指定されている固有のカスタマイズ対象処理を求める。ここで、カスタマイズ対象

処理とは、処理の流れが特定の処理パターンに合致したとき実行される具体的な処理のことであり、例えば処理パターンが“マッチング”であれば、キー値がマッチしたとき行うマッチ処理とキー値がマッチしないとき行うアンマッチ処理のいずれか一方または両方がカスタマイズ対象処理となり得る。最後に、ステップ６０２で取得したサブクラスのオブジェクト定義の雛型７００の該当処理に対応するメソッド記述に、求めたカスタマイズ対象処理の記述７１１を設定し、これによって固有の処理（マッチ処理）を実現するためのオブジェクト定義文及び処理命令文を有するソースコードを生成する（ステップ６０４）。

【００２８】図８は、図１中のソースファイル出力処理部２Ｄによる処理の流れを説明するフローチャートである。同図中、最初に入力部１で入力された仕様関連情報から生成情報１ａを取得する（ステップ８０１）。ここで、生成情報１ａは、例えば生成先ディレクトリ・ファイル名称である。続いて、生成情報１ａ中のクラス名称及びプログラム名称を、ファイル／ＤＢ情報処理部２Ａ，パターン情報処理部２Ｂ，カスタマイズ情報処理部２Ｃで先に生成されたソースコード中のCLASS-ID及びPROGRAM-IDにそれぞれ設定する（ステップ８０２）。また、これらのCLASS-ID及びPROGRAM-IDを用いて、各ソースコードを格納するソースファイルのファイル名称を組み立てる（ステップ８０３）。最後に、組み立てたファイル名称をそれぞれ付加したソースファイルに、対応するソースコードを出力する（ステップ８０４）。

【００２９】図９は、図１のシステムによって生成されたソースコードを含む複数のソースファイルの一例を示す図であり、作業ファイルや中間ファイルに直接アクセスを行うアプリケーションを構成するソースファイル群を表す。同図中、生成されたソースファイルは、所定の入出力処理を逐次アクセスによって実行するためのオブジェクト定義文及び処理命令文を有するソースコードすなわちファイル情報管理クラスを含むソースファイル９１，新たなアプリケーションによる処理手順を表す処理パターンに対応するオブジェクト定義文及び処理命令文を有するソースコードすなわちパターン処理管理クラスを含むソースファイル９２，新たなアプリケーションに固有の処理を実現するためのオブジェクト定義文及び処理命令文を有するソースコードすなわちカスタマイズ処理管理クラスを含むソースファイル９３，これらのソースファイル９１～９３をまとめるプログラム定義の処理命令文を有するソースコードすなわちメインプログラムを含むソースファイル９４の合計４種類のソースファイルに大別される。

【００３０】図１０は、図１のシステムによって生成されたソースコードを含む複数のソースファイルの他の例を示す図であり、作業ファイルや中間ファイルに直接アクセスを行わないアプリケーションを構成するソースファイル群を表す。同図中、生成されたソースファイルは、所定の入出力処理を逐次アクセスによることなく実行するためのオブジェクト定義文及び処理命令文を有するソースコードすなわちコレクションクラスを含むソースファイル１０１及び要素クラスを含むソースファイル１０２，新たなアプリケーションによる処理手順を表す処理パターンに対応するオブジェクト定義文及び処理命令文を有するソースコードすなわちパターン処理管理クラスを含むソースファイル１０３，新たなアプリケーションに固有の処理を実現するためのオブジェクト定義文及び処理命令文を有するソースコードすなわちカスタマイズ処理管理クラスを含むソースファイル１０４，これらのソースファイル１０１～１０４をまとめるプログラム定義の処理命令文を有するソースコードすなわちメインプログラムを含むソースファイル１０５の合計５種類のソースファイルに大別される。

【００３１】以上のように、本実施形態のソースコードの自動生成システムによれば、新たなアプリケーション開発に際し、この新たなアプリケーションで再利用しようとする従来の構造化プログラミングの手法で得られた“パターン”及び“部品”を特定する仕様関連情報、すなわち生成情報１ａ，入出力ファイル／ＤＢ定義情報１ｂ，処理情報１ｃ，カスタマイズ情報１ｄの各情報を適切に決めて入力部１に入力することにより、該当する“パターン”及び“部品”に対応する雛型が雛型データベース３から取得され、“パターン”すなわち処理パターンについては、パターン情報処理部２Ｂによりその決定に必要な不足情報の入力要求が行われるとともにカスタマイズ情報処理部２Ｃによって新たなアプリケーションに固有のカスタマイズ対象処理の記述が雛型に設定される。このため、従来の構造化プログラミングの手法で得られた“パターン”及び“部品”とオブジェクト指向プログラミングの手法で得られた“オブジェクト”とを意図的に区別する必要がなくなって同様に扱うことができ、これまで構造化プログラミングの手法によるプログラム開発で得られた既存のプログラム資産とオブジェクト指向プログラミングとの融合を容易に図ることができる。

【００３２】また、作業ファイルや中間ファイルに逐次アクセスすることなく入出力処理を実現するオブジェクト定義及びこのオブジェクト定義を扱う所定のプログラム定義からなる雛型をあらかじめ雛型データベース３中に蓄積しておくことにより、図１０に示したように、所定の入出力処理を逐次アクセスによることなく実行するためのオブジェクト定義文及び処理命令文を有するソースコードすなわちコレクションクラスを含むソースファイル１０１及び要素クラスを含むソースファイル１０２の生成が可能となる。このため、永続オブジェクトとして認識されない作業ファイルや中間ファイルに関連する“オブジェクト”によって行われる無駄なアクセスが抑

止され、処理効率の低下や永続オブジェクトの認識のあいまいさの発生を防止することができる。

【００３３】なお、上述した雛型データベース３を保持するデータベースファイルと雛型設定処理部２を実現させるプログラムファイルとは、例えば、フロッピーディスクやＣＤ－ＲＯＭなどの情報記憶媒体に保管された形で提供することができる。この場合、同一の情報記憶媒体に両者を一括して保管しても、異なる情報記憶媒体に両者を別個に保管しても、どちらでもよい。

【００３４】

【発明の効果】以上詳しく説明したように、本発明のソースコードの自動生成システムによれば、従来の構造化プログラミングの手法で得られた“パターン”及び“部品”とオブジェクト指向プログラミングの手法で得られた“オブジェクト”とを意図的に区別する必要がなくなって同様に扱うことができ、これまで構造化プログラミングの手法によるプログラム開発で得られた既存のプログラム資産とオブジェクト指向プログラミングとの融合を容易に図ることができる。

【００３５】また、逐次アクセスなしに入出力処理を実現する“オブジェクト”の雛型をあらかじめ雛型データベース中に蓄積しておけば、永続オブジェクトとして認識されない作業ファイルや中間ファイルに関連する“オブジェクト”によって行われる無駄なアクセスが抑止され、処理効率の低下や永続オブジェクトの認識のあいまいさの発生を防止することができる。

【図面の簡単な説明】

【図１】本発明のソースコードの自動生成システムの一実施形態を示すブロック図である。

【図２】図１中のファイル／ＤＢ情報処理部による処理の流れを説明するフローチャートである。

【図３】図２の処理によって設定されるソースコードの＊雛型の一例を示す図である。

【図４】図１中のパターン情報処理部による処理の流れを説明するフローチャートである。

【図５】図４の処理によって設定されるソースコードの雛型の一例を示す図である。

【図６】図１中のカスタマイズ情報処理部による処理の流れを説明するフローチャートである。

【図７】図６の処理によって設定されるソースコードの雛型の一例を示す図である。

【図８】図１中のソースファイル出力処理部による処理の流れを説明するフローチャートである。

【図９】図１のシステムによって生成されたソースコードを含む複数のソースファイルの一例を示す図である。

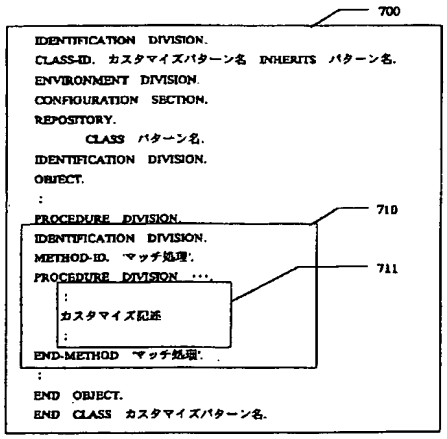
【図１０】図１のシステムによって生成されたソースコードを含む複数のソースファイルの他の例を示す図である。

【符号の説明】

１　入力部
２　雛型設定処理部
３　雛型データベース
２Ａ　ファイル／ＤＢ情報処理部
２Ｂ　パターン情報処理部
２Ｃ　カスタマイズ情報処理部
２Ｄ　ソースファイル出力処理部
２ｔ　認識された処理パターン
３Ａ　入出力操作に関連したソースコードの雛型群
３Ｂ　アプリケーション全体の処理手順に関連したソースコードの雛型群
３Ｃ　アプリケーション各所で実行される処理方式に関連したソースコードの雛型群
４　生成されたソースコードを含むソースファイル群

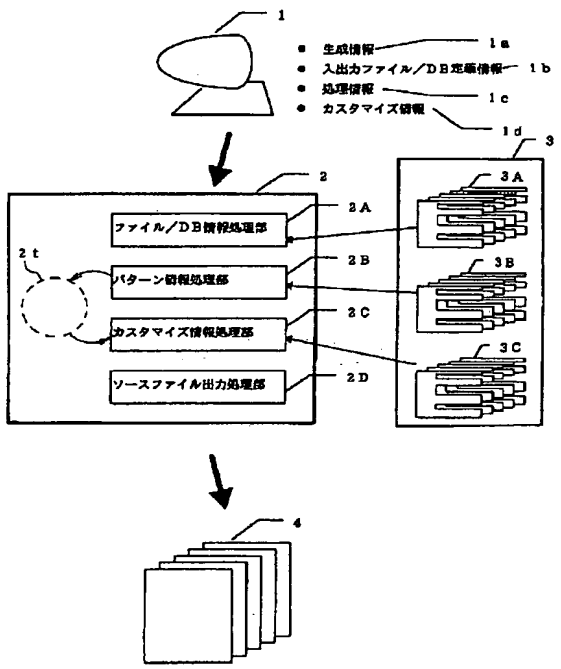
【図７】

【図７】



【図１】

【図１】



【図３】

【図３】

```
300
IDENTIFICATION DIVISION.
CLASS-ID. ファイル名 INHERITS FILEF.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
        CLASS FILEF.
IDENTIFICATION DIVISION.
OBJECT.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.                                   310
SELECT [ファイル名]                              311
    ASSIGN TO FILENAME
    ACCESS MODE DYNAMIC
    ORGANIZATION [INDEXED]                      312
    RECORD KEY [キー項目]                        313
DATA DIVISION.
FILE SECTION.                                   320
FD [ファイル名].                                 321
01 [レコード名].                                 322
[レコード定義]                                   323
:
PROCEDURE DIVISION.
IDENTIFICATION DIVISION.
METHOD-ID. 'OPENF'.
PROCEDURE DIVISION.                             330
        OPEN 入出力区分 [ファイル名]              331
END-METHOD 'OPENF'.
:
END OBJECT.
END CLASS ファイル名.
```

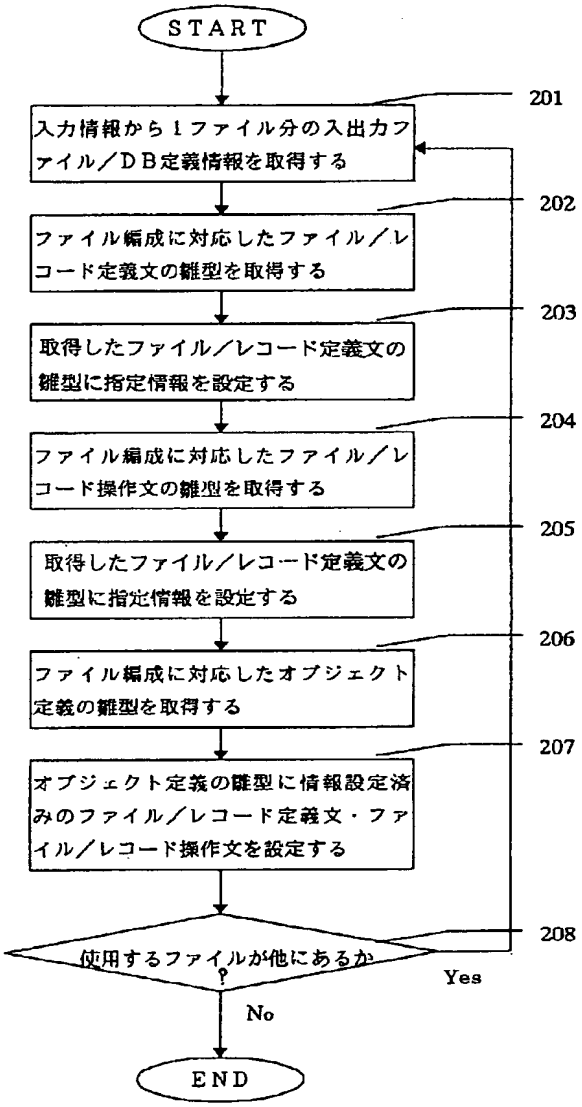
【図５】

【図５】

```
500
IDENTIFICATION DIVISION.
CLASS-ID. パターン名 INHERITS HCB-OBJECT.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
        CLASS HCB-OBJECT.
IDENTIFICATION DIVISION.
OBJECT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ファイル名-HDL. OBJECT REFERENCE FILEF.
:
PROCEDURE DIVISION.
IDENTIFICATION DIVISION.
METHOD-ID. 'マッチング'.                        511
PROCEDURE DIVISION.                             510
    :
    IF XXX = YYY
        THEN
            INVOKE SELF 'マッチ処理' ...
        ELSE
            INVOKE SELF 'アンマッチ処理' ...
    END-IF.
    :
END-METHOD 'マッチング'.
IDENTIFICATION DIVISION.
METHOD-ID. 'マッチ処理'.
PROCEDURE DIVISION ....
END-METHOD 'マッチ処理'.
:
END OBJECT.
END CLASS パターン名.
```

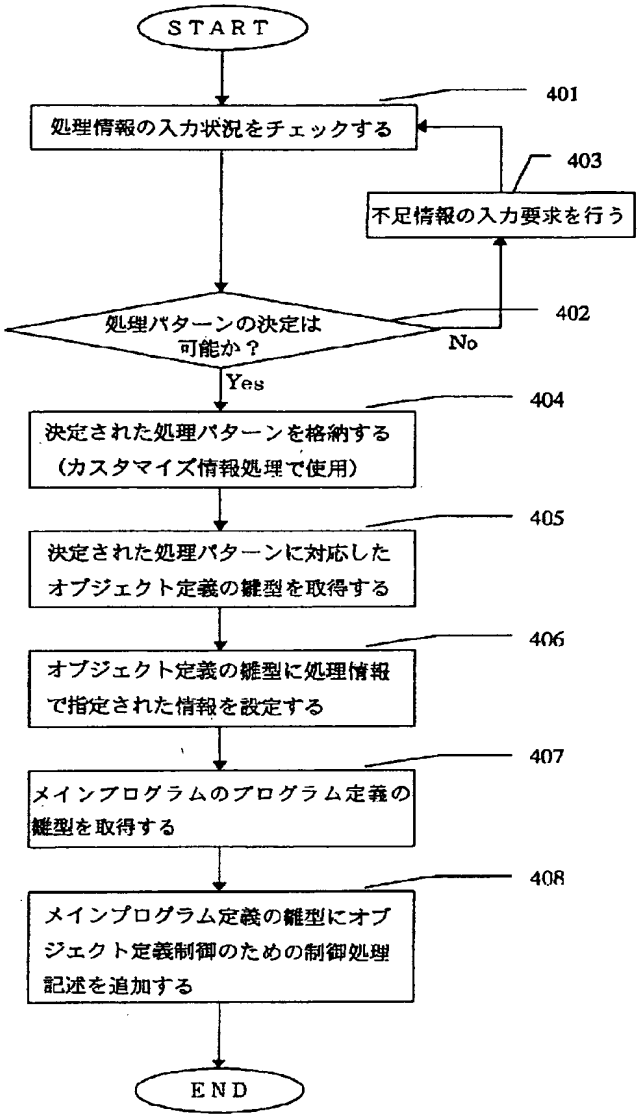
【図２】

【図２】


START
201
入力情報から１ファイル分の入出力ファイル／ＤＢ定義情報を取得する
202
ファイル編成に対応したファイル／レコード定義文の雛型を取得する
203
取得したファイル／レコード定義文の雛型に指定情報を設定する
204
ファイル編成に対応したファイル／レコード操作文の雛型を取得する
205
取得したファイル／レコード定義文の雛型に指定情報を設定する
206
ファイル編成に対応したオブジェクト定義の雛型を取得する
207
オブジェクト定義の雛型に情報設定済みのファイル／レコード定義文・ファイル／レコード操作文を設定する
208
使用するファイルが他にあるか？
Yes
No
END

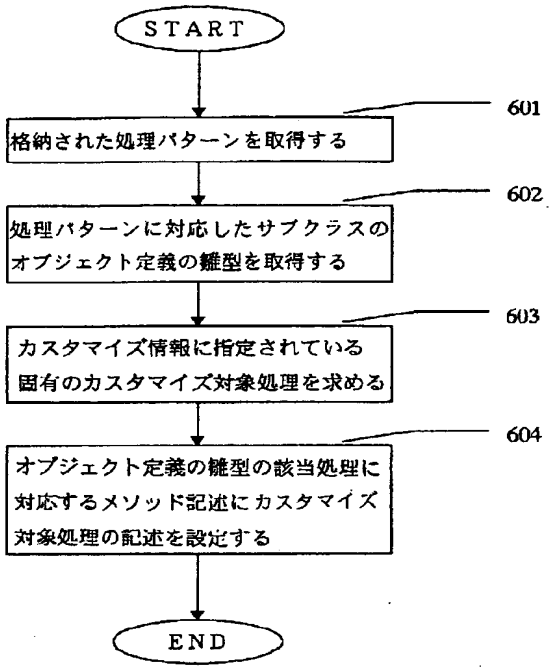
【図４】

【図４】


START
401
処理情報の入力状況をチェックする
403
不足情報の入力要求を行う
処理パターンの決定は可能か？
402
No
Yes
404
決定された処理パターンを格納する（カスタマイズ情報処理で使用）
405
決定された処理パターンに対応したオブジェクト定義の雛型を取得する
406
オブジェクト定義の雛型に処理情報で指定された情報を設定する
407
メインプログラムのプログラム定義の雛型を取得する
408
メインプログラム定義の雛型にオブジェクト定義制御のための制御処理記述を追加する
END

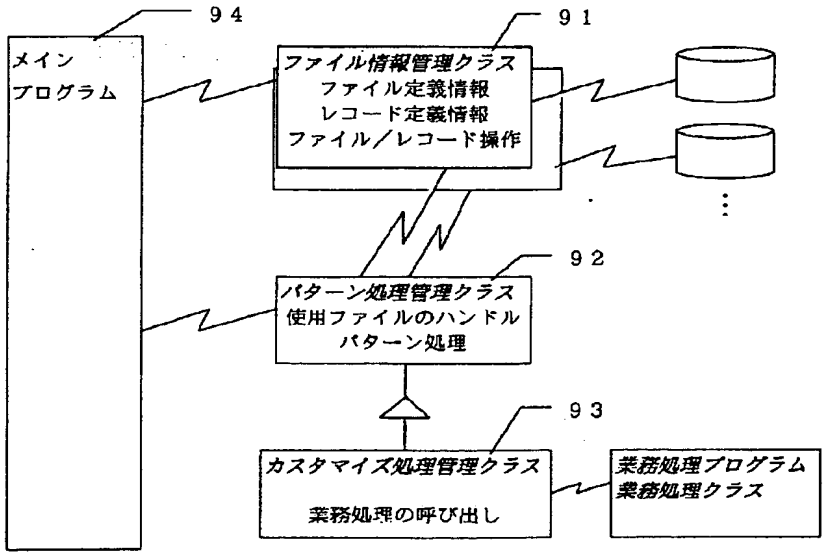
【図６】

【図６】


START
601
格納された処理パターンを取得する
602
処理パターンに対応したサブクラスの
オブジェクト定義の雛型を取得する
603
カスタマイズ情報に指定されている
固有のカスタマイズ対象処理を求める
604
オブジェクト定義の雛型の該当処理に
対応するメソッド記述にカスタマイズ
対象処理の記述を設定する
END

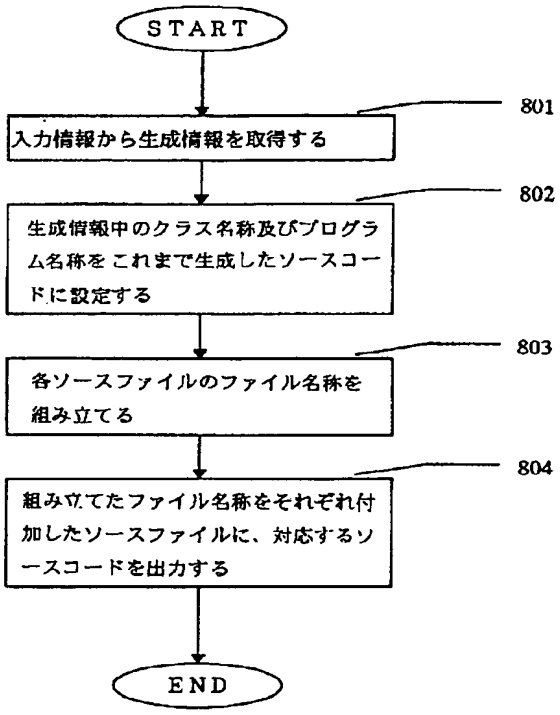
【図９】

【図９】


９４
メイン
プログラム
９１
ファイル情報管理クラス
ファイル定義情報
レコード定義情報
ファイル／レコード操作
９２
パターン処理管理クラス
使用ファイルのハンドル
パターン処理
９３
カスタマイズ処理管理クラス
業務処理の呼び出し
業務処理プログラム
業務処理クラス

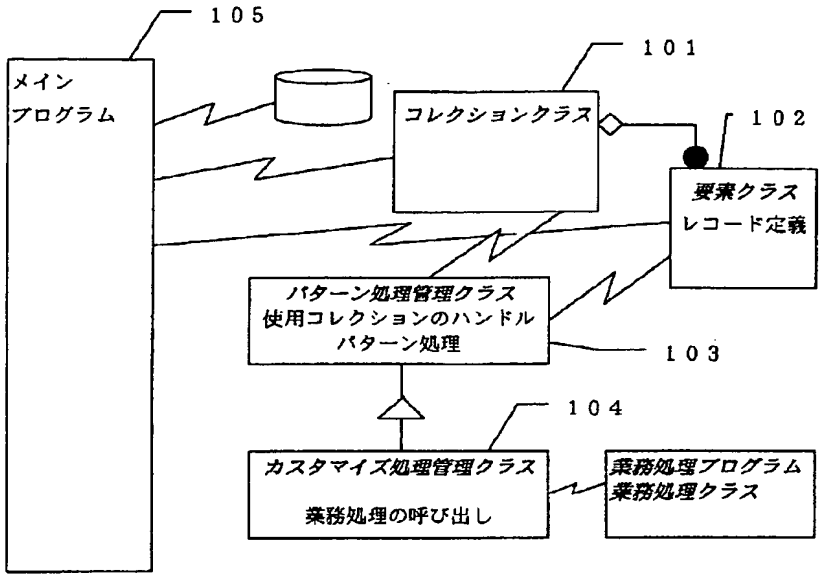
【図８】

【図８】


START
801
入力情報から生成情報を取得する
802
生成情報中のクラス名称及びプログラム名称をこれまで生成したソースコードに設定する
803
各ソースファイルのファイル名称を組み立てる
804
組み立てたファイル名称をそれぞれ付加したソースファイルに、対応するソースコードを出力する
END

【図１０】

【図１０】


１０５
メイン
プログラム
１０１
コレクションクラス
１０２
要素クラス
レコード定義
パターン処理管理クラス
使用コレクションのハンドル
パターン処理
１０３
１０４
カスタマイズ処理管理クラス
業務処理の呼び出し
業務処理プログラム
業務処理クラス